JCSI 1 (2016) 34-37
Journal of Computer Sciences Institute

Wysłane: 2016-10-03
Przyjęte: 2016-10-12

Zbadanie wydajności aplikacji internetowych utworzonych z wykorzystaniem Spring MVC oraz JavaServer Faces

Nazar Kovalchuk*, Edyta Łukasik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Celem artykułu było porównanie wybranych szkieletów programistycznych w technologii Java. Do analizy zostały wybrane Spring MVC oraz JavaServer Faces. Została przeprowadzona analiza teoretyczna związana z funkcjonowaniem badanych frameworków. Dla porównania wybranych szkieletów programistycznych były utworzone dwie aplikacje, posiadające podobną funkcionalność. wyniki przeprowadzonych badań dotyczących wydajności oraz pomiaru metryk kodu utworzonych aplikacji zostały pokazane w postaci wykresów.

Słowa kluczowe: Spring MVC; JavaServer Faces; Java; aplikacja internetowa

*Autor do korespondencji.

Adres E-mail: koval014051993@gmail.com

Efficiency test of web applications created using Spring MVC and JavaServer Faces frameworks

Nazar Kovalchuk*, Edyta Łukasik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The purpose of the article was to compare selected frameworks for web applications Java technology. Spring MVC and JavaServer Faces were selected for the analysis. Theoretical analysis related with the functioning of the examined frameworks was conducted. For comparison of selected frameworks there were created two applications that have similar functionality. Performance testing and measuring code metrics of created applications were conducted.

Keywords: Spring MVC; JavaServer Faces; Java; web application

*Corresponding author.

E-mail address: koval014051993@gmail.com

1. Wstęp

Celem artykułu jest porównanie wybranych szkieletów programistycznych w technologii Java. Do analizy zostały wybrane frameworki Spring MVC oraz JavaServer Faces. Analiza obejmuje teoretyczne aspekty ich funkcjonowania oraz wykorzystanie w praktyce.

W celu utworzenia aplikacji internetowych wykorzystano następujące narzędzia:

- szkielety programistyczne Spring MVC oraz JavaServer Faces;
- zintegrowane środowisko programistyczne NetBeans [1] oraz Eclipse [2];
- server MySQL [3];
- Maven – narzędzie do budowy projektu w technologii Java: kompilacji, tworzenia jar, generowania dokumentacji[4].

Dla porównania Spring MVC oraz JSF została przeprowadzona analiza teoretyczna związana z funkcjonowaniem badanych szkieletów programistycznych, która wskazała podobieństwa oraz różnice w działaniu obu ram projektowych. Omawiając równocześnie mocne i słabe strony każdego ze szkieletów w sytuacjach typowych dla działania aplikacji internetowych, w analizie zwrócono uwagę na najważniejsze aspekty związane z działaniem badanych frameworków, czyli:

- architektura;
- model programistyczny;
- walidacja danych;
- internacjonalizacja.

Jedną z najlepszych metod oceny wydajności jest wykonanie testów z 1 aktywnym użytkownikiem, 2 użytkownikami, 4 użytkownikami, 8 użytkownikami itp[5].

Najważniejszymi parametrami, które należy rejestrować w każdym teście wydajności, są przepustowość (liczba żądań na sekundę) oraz czas odpowiedzi.

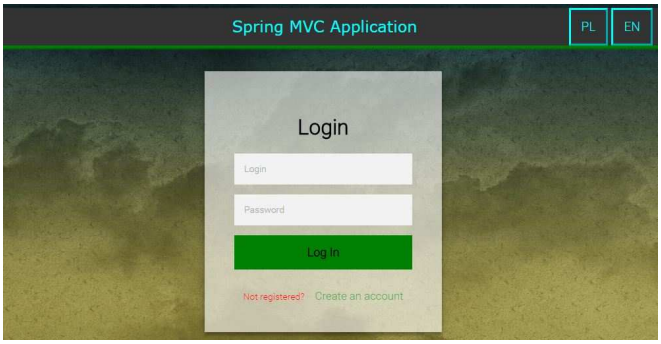
2. Opis aplikacji

Utworzone zostały dwie aplikacje oparte o: Spring MVC oraz JavaServer Faces. Każda z nich posiada następujące funkcjonalności:

- logowanie użytkownika;
- rejestracja użytkownika;
- użytkownicy aplikacji posiadają dwie role (admin oraz prosty użytkownik);

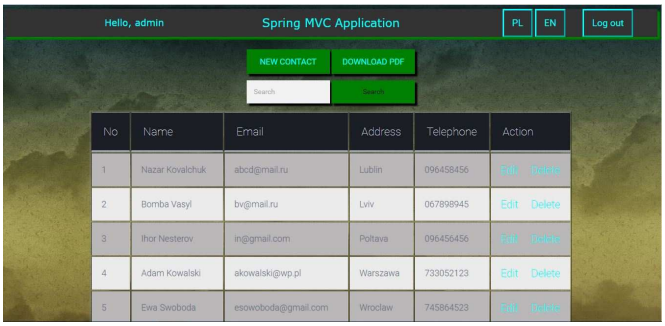
- dodawanie nowego kontaktu;
- usuwanie oraz edycja wybranego kontaktu;
- wyszukiwanie kontaktów;
- pobranie listy kontaktów w formacie PDF oraz Excel;
- walidacja wprowadzonych danych;
- zmiana języka aplikacji (angielski oraz polski).

Wygląd przykładowych ekranów działających aplikacji został pokazany na rysunkach 1, 2 i 3. Po uruchomieniu aplikacji pojawia się okno logowania użytkownika przedstawione na rys. 1.



Rys. 1. Logowanie użytkownika w aplikacji Spring MVC

Przy zalogowaniu użytkownika jako administratora pojawia się główne okno aplikacji jak na rys.2.



Rys. 2. Główne okno aplikacji Spring MVC

Aplikacja posiada wybór języka: polski lub angielski. Na rysunku 3 przedstawiono wygląd aplikacji JSF w języku polskim.

Rys. 3. Zmiana języka aplikacji JavaSever Faces

3. Badania wydajności

Do zbadania wydajności utworzonych aplikacji zostało wykorzystane narzędzie JMeter.

W celu przeprowadzenia testów wydajności były utworzone dwa scenariusze testów. Pierwszy scenariusz obejmował operacji odczytu danych, czyli akcji, które nie mogli mieć wpływ na stan aplikacji.

Drugi scenariusz obejmował operacji, które wpływali na stan aplikacji. Do operacji takiego typu można odnieść dodawanie nowych użytkowników, modyfikacje kontaktów oraz ich usuwanie.

Testy dotyczące wydajności zostały przeprowadzone na obu badanych aplikacjach opartych o Spring MVC oraz JavaServerFaces.

Dane testy były przeprowadzone dla różnej ilości użytkowników od 1 do 512.

Wyniki pierwszego testu aplikacji opartej o Spring MVC są przedstawione w tabeli 1.

Tabela 1. Wydajność aplikacji opartej o Spring MVC przy pierwszym teście

Liczba użytkowników	Liczba żądań/sek	Czas odpowiedzi [ms]
1	60	13
2	111	15
4	137	24
8	164	40
16	174	72
32	189	155
64	148	413
128	146	832
256	136	1509
512	126	2471

Wyniki drugiego testu aplikacji opartej o Spring MVC są przedstawione w tabeli 2.

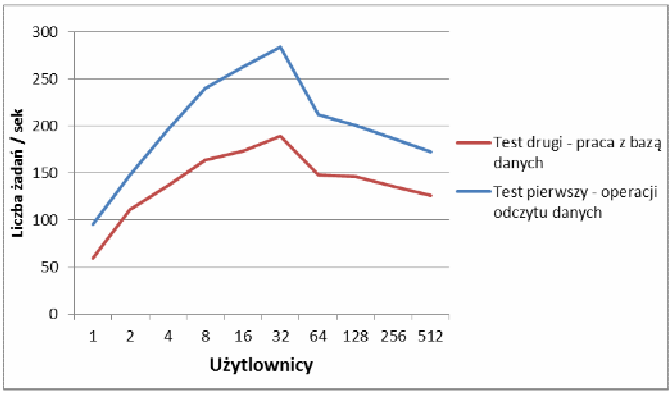
Tabela 2. Wydajność aplikacji opartej o Spring MVC przy drugim teście

Liczba użytkowników	Liczba żądań/sek	Czas odpowiedzi [ms]
1	96	2
2	148	2
4	196	3
8	240	6
16	263	9
32	284	10
64	212	12
128	201	13
256	187	303
512	173	1530

Na podstawie otrzymanych wyników został utworzony wykres przedstawiony na rys. 4 dla porównania wydajności aplikacji Spring MVC przy pierwszym i drugim teście.

Jak widać na podsdtawi ewykresu przedstawionego na rys.4 przy drugim teście, który dotyczył pracy z bazą danych,

wydajność aplikacji spadła, co jest oczekiwanym wynikiem, ponieważ na takie operacje aplikacja potrzebuje więcej zasobów.



Rys. 4. Wydajność aplikacji opartej o Spring MVC przy pierwszym i drugim teście.

Wyniki pierwszego testu aplikacji opartej o JavaServer Faces są przedstawione w tabeli 3.

Tabela 3. Wydajność aplikacji opartej o JavaServer Faces przy pierwszem teście

Liczba użytkowników	Liczba żądań/sek	Czas odpowiedzi [ms]
1	19	25
2	35	29
4	57	34
8	94	41
16	109	72
32	112	140
64	103	306
128	96	651
256	94	1435
512	86	2755

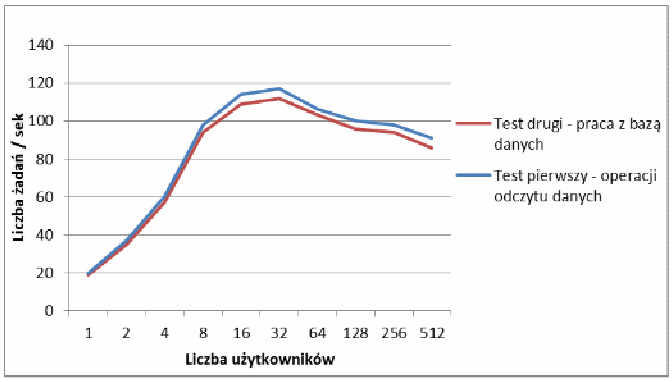
Wyniki drugiego testu aplikacji opartej o JavaServer Faces są przedstawione w tabeli 4.

Tabela 4. Wydajność aplikacji opartej o JavaServer Faces przy drugim teście

Liczba użytkowników	Liczba żądań/sek	Czas odpowiedzi [ms]
1	20	27
2	37	27
4	60	33
8	98	41
16	114	70
32	117	140
64	106	304
128	100	651
256	98	1289
512	91	2821

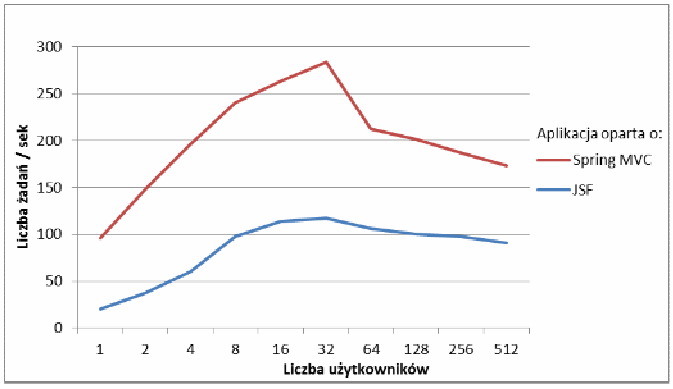
Na podstawie otrzymanych wyników został utworzony wykres przedstawiony na rys. 5 dla porównania wydajności aplikacji JavaServer Faces przy pierwszym i drugim teście.

Jak widać na wykresach pokazanym na rys. 5 wyniki obu testów aplikacji opartej o JavaServer Faces są bardzo podobne. Wynika to z wielkości przesyłanego ciasteczka (około 5 KB), które przy zwiększonej liczbie żądań zaczyna odgrywać znaczną rolę.

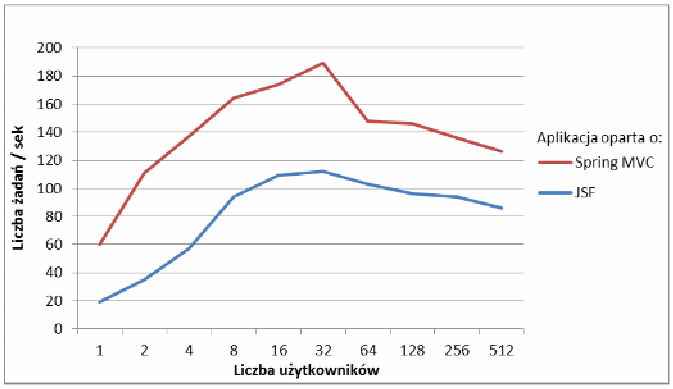


Rys. 5. Wydajność aplikacji opartej o JavaServer Faces przy pierwszym i drugim teście

Dla porównania wydajności aplikacji Spring MVC oraz JavaServer Faces były zbudowane wykresy na podstawie otrzymanych wyników, które pokazano na rys. 6 i 7.



Rys. 6. Porównanie wydajności aplikacji Spring MVC oraz JavaServer Faces przy pierwszym teście



Rys. 7. Porównanie wydajności aplikacji Spring MVC oraz JavaServer Faces przy drugim teście

Z otrzymanych wyników przedstawionych na rys. 6 i 7 można wnioskować, że aplikacja oparta o Spring MVC jest bardziej wydajna, niż aplikacja oparta o JavaServer Faces. Wydajność aplikacji liniowo narasta w przypadku zmiany liczby klientów od 1 do 32. Jednak w przedziale od 32 do 512 klientów, wzrost wydajności zaczyna powoli spadać. Punkt ten nazywa się punktem nasycenia. To jest kluczowe znaczenie,

które trzeba znaleźć podczas testów wydajności. Punkt nasycenia mówi o tym, że osiągnięto najwyższą wydajność dla aplikacji. Wzrost liczby użytkowników powyżej tej wartości tylko wydłuży czas reakcji, nie zwiększając ogólnej wydajności aplikacji.

4. Metryki kodu

Celem pomiaru metryk kodu było porównanie wielkości zaimplementowanych aplikacji opartych o Spring MVC oraz JavaServer Faces.

Pomiar metryk kodu został przeprowadzony za pomocą wtyczki środowiska Eclipse o nazwie Metrics oraz wtyczki środowiska NetBeans o nazwie SourceCodeMetrics. Wyniki pokazano w tabeli 5.

Tabela 5. Metryki kodu aplikacii opartych o Spring MVC oraz JavaServer Faces

Metryka	Spring MVC	JavaServer Faces
Liczba linii kodu	568	508
Liczba klas	7	5
Liczba pól	14	23
Liczba metod	69	45
Liczba linii kodu metod	257	236
Liczba pakietów	5	5

Zmierzona została liczba linii kodu, artefaktów oraz elementów klas. W analizie uwzględniono klasy DAO odpowiedzialne za dostęp do danych oraz logikę biznesową aplikacji z wykorzystaniem szkieletów Spring MVC oraz JavaServer Faces. W przypadku linii kodu zostały zmierzone dwie metryki. Pierwsza z nich jest związana z całkowitą liczbą linii kodu całej aplikacji, druga dotyczy wyłącznie kodu umieszczonego w metodach. Aplikacja oparta o JavaServer Faces zawiera mniej linii kodu niż aplikacja oparta o Spring MVC.

Tabela 6 przedstawia porównanie liczby linii kodu podzielonego według funkcjonalności. Wprowadzono podział na kod kontrolerów, kod walidacji, kod odpowiedzialny za internacjonalizację oraz kod dostępu do bazy danych (DAO).

Tabela 6. Metryki kodu aplikacii opartych o Spring MVC oraz JavaServer Faces

Liczba linii kodu	Spring MVC	JavaServer Faces
Kontrolery	146	52
Walidacja	20	-
Internacjonalizacja	22	24
DAO	140	210
Liczba linii kodu metod	257	236
Liczba pakietów	5	5

Aplikacja oparta o JavaServer Faces nie posiada w kodzie logiki związanej z walidacją wprowadzanych danych. Wszystkie aspekty walidacji wprowadzanych danych udało się zaimplementować na stronach internetowych.

W przypadku kontrolerów aplikacja wykorzystująca Spring MVC posiada więcej kodu, ponieważ kontrolery zawierają metody walidacji wprowadzonych danych.

Ilość kodu odpowiedzialnego za internacjonalizację jest podobna w obu aplikacjach opartych o Spring MVC oraz o JavaServer Faces.

W przypadku klas odpowiedzialnych za dostęp do bazy danych aplikacja wykorzystująca Spring MVC jest łatwiejsza. Spring MVC posiada gotową klasę JdbcTemplate, która pozwala zmniejszyć ilość kodu.

5. Wnioski

Celem badań było porównanie dwóch wybranych szkieletów programistycznych w technologii Java. Analizy została przeprowadzona dla Spring MVC oraz JavaServer Faces. Przedstawione w artykule porównanie miało za zadanie ułatwienie programiście dokonanie wyboru między dwoma alternatywnymi rozwiązaniami.

Przeprowadzona analiza teoretyczna związana z funkcjonowaniem badanych szkieletów programistycznych wskazała podobieństwa oraz różnice w ich działaniu.

Spring MVC realizuje koncepcję aplikacji internetowej opartej o żądania i odpowiedzi [6]. Używa do tego kontrolerów, działających na modelu. JavaServer Faces umożliwia tworzenie aplikacji internetowych z naciskiem na interfejs użytkownika zbudowany w sposób komponentowy [7].

Przeprowadzone zostały badania wydajności oraz pomiar metryk kodu utworzonych aplikacji. Badania wydajności zostały wykorzystane do oceny zachowania aplikacji w sytuacji dużego obciążenia, co jest ważną kwestią w przypadku aplikacji dostępnych dla szerokiego grona osób. Aplikacja oparta o Spring MVC jest bardziej wydajnościowa, niż aplikacja oparta o JavaServer Faces.

Pomiar metryk kodu wskazał, że aplikacja wykorzystująca Spring MVC jest większa ze względu na liczbę linii kodu oraz artefaktów programistycznych.

Literatura

[1] https://netbeans.org/features/index_ru.html [14.06.2016].
[2] http://hightech.in.ua/content/art-eclipse-platform [08.06.2016].
[3] http://danneo.ru/coding/read-mysql-a-control-system-of-databases.html [17.06.2016].
[4] https://maven.apache.org/what-is-maven.html [16.06.2016].
[5] http://www.ibm.com/developerworks/ru/library/wes-1208_hare/ [21.07.2016].
[6] Amuthan G.: Spring MVC: przewodnik dla początkujących. Gliwice: Wydawnictwo Helion, cop. 2015.
[7] [David Geary, Cay S. Horstmann: JavaServer Faces. Gliwice : Wydawnictwo Helion, cop. 2011.